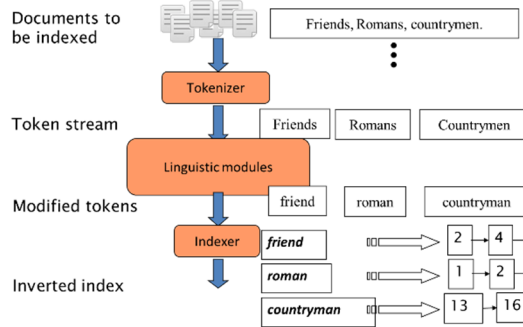


אחזור מידע

Document ingestion

כאמור, אנו עובדים עם מסמכים. אם נחזור ונסתכל על תהליך יציאת ה-Inverted Index, הרי שהתהליך מתחיל עם פרוק המסמכים למילים. ובכן, מהו בדיק מסמך ?



עד עכשיו התייחסנו למסמך כאוסף של מילים. אולם בהרבה מקרים זה, לא כך. במסמכים רבים אוסף המילים שלנו "מוחבאים" בין נתונים נוספים. בחלק מהמסמכים הם שמורים בצורה "מוצפנת" או "מעורבלת", ולכן יתכן ויהיה שלב מקדים והוא המרה של המסמך מפורמט אחד לפורמט שני.

אם כך, מהן הפעולות, או מהם האתגרים שאיתם אנו צריכים להתמודד בשלב המקדים.

- פורמט של המסמך אותו אנו מנתחים. כאמור, המסמכים מגיעים בפורמטים שונים, כגון, pdf, word, excel, html, וכו'. הפורמטים השונים "מקודדים" את המילים בצורות שונות, ואנו צריכים לחלץ את המילים מתוך הפורמט של המסמך.
- השפה בה כתוב המסמך. אנו צריכים לזהות את השפה בה כתוב המסמך, מכיוון שהן ה-Tokenizer והן ה-Linguistic modules תלויים בה.
- אוסף התווים (קידוד) שבו נעשה שימוש במסמך. כידוע, ישנם סוגים שונים של אוספי תווים, למשל CP1525, UTF8, וכדומה. אוסף התווים בו נעשה שימוש קובע למעשה איך אנו מתרגמים תו (או אוסף תווים) לאותיות, ולכן חשוב לנו לדעת מהו אוסף התווים שבו נעשה שימוש, על מנת לקבל את האותיות והמילים הנכונות.

כל אחד מהבעיות הללו היא בעיה סיווג, אשר נלמד בהמשך הקורס. ברור המקרים, הטיפול בבעיות אלו נעשה ע"י שימוש בהיוריסטיקות שונות (למשל, ע"י שימוש בסיומת הקובץ, או בחינה של meta-data של הקובץ).

האתגרים ההלו יכולים להיות מסובכים או קשים ביותר במספר מקרים. להלן מס' דוגמאות.

- המסמכים אותם אנו מאנדקסים יכולים להיות בשפות שונות. המשמעות היא שקובץ האינדקס שלנו יכיל מושגים משפות שונות.
 - יתכן שקובץ (או קובץ + מרכיבים נוספים, כגון מסמכי השלמה שונים וכו') יכיל שפות שונות ו/או פורמטים שונים. למשל, קובץ אי-מייל בעברית עם קובץ מצורף באנגלית. מסמך וורד בעברית המכיל ציטוט באנגלית.
- קיימות היום ספריות קוד מוכנות, הן מסחריות והן חופשיות, המסוגלות להתמודד עם הבעיות הללו.

עד עתה השתמשנו במושג מסמך. אנו מאנדקסים מסמכים. אנו מחפשים מסמכים. מהו מסמך ?

- האם זה קובץ ?
- האם זה אי-מייל ? אם כן, איך נתייחס לאי-מייל אשר מצורפים אליו קבצים נוספים. תוכנות מסוימות שומרות ספרייה (למשל את ה-Inbox) עם כל המיילים שבו כקובץ אחד - איך להתייחס לכך ?
- האם זה אוסף של קבצים (למשל עבודה הכתובה בקובץ וורד ומחולקת למספר קבצים) ?
- ספר - האם נתייחס לספר כולו כמסמך אחד, או שאולי, נתייחס לכל פרק כמסמך אחד (נניח שיש לנו ספר על ההיסטוריה של אירופה בימי הביניים. הספר מדבר בין היתר בפרק אחד על הנצרות, ובפרק אחר על הקמת האוניברסיטאות. שאילתא כגון Christ university, יכולה להחזיר את הספר הזה מכיוון ששני המושגים נמצאים

בו. אבל אם נתייחס לכל פרק כמסמך בנפרד, אז לא נקבל את הספר כתשובה לשאלתא). מצד שני, חלוקה קטנה מידי עלולה לגרום לאיבוד מידע.

Tokens

אחרי שהגדרנו מהו מסמך. עברנו על המסמכים שלנו, והוצאנו מהם את המילים, ונוכלים להתחיל בתהליך ה-Tokenization.

Token הוא רצף של תווים במסמך שלנו (שיכול לכלול רווחים וסימני פיסוק). אם רצף תווים מסוים חוזר פעמיים במסמך, ונוקבל את אותו ה-token פעמיים. במילון שלנו ונו לא מוצאים את ה-tokens שהוצאנו מהמסמכים, אלא את ה-terms (ביטויים / מושגים). Term הוא token אשר עבר עיבוד מסוים, ולרוב קיבל צורה חדשה כפי שנראה בהמשך.

כביכול, עבודתו של ה-tokenizer היא פשוטה, חלוקה של המסמך למילים, אבל לא תמיד זה כך. להלן מס' נושאים שיש להתייחס עליהם בעת הטוקניזציה. נתון לנו הביטוי "Finland's capital". כיצד נתייחס למילה "Finland's"? האם כשתי מילים, Finland ו-s, או Finlands או Finland's? כלומר, איך נתייחס במקרה הזה לגרש שמופיע במילה. באופן דומה, איך נתייחס לגרש בטוקן aren't, הרי כאן המשמעות היא שונה, לקבל צורה מקוצרת ל-are not.

דוגמה נוספת, הביטוי "Hewlett-Packard". במקרה הזה, בין שתי המילים מפריד מקף, ואנו צריכים לקבוע איך נתייחס למקף הזה. האם נפרק אותו ל-Hewlett ו-Packard כשתי מילים נפרדות? במקרה הזה, "Hewlett-Packard" הינו שם של חברה, המורכבת משמות שני המייסדים שלה. אם נפרק את הביטוי לשתי מילים, ונו עלולים להחזיר מסמכים שבהם שתי המילים מופיעות בנפרד, כשני שמות שונים, ולא כשם החברה (למשל, אם יש מסמך שמתאר את הביוגרפיה של Hewlett, כנראה שכתוב שם שהוא עבד או הקים חברה ביחד עם Packard). באופן דומה, מה יקרה עם המושגים lower-case או co-education. נניח ובשאלתא המשתמש מקליד "Packard Hewlett" (ללא הרווח), איך נדע למה התכוון?

San Francisco, במקרה זה אין מקף בין שתי המילים. האם ונו צריכים להתייחס אליהן כטוקן אחד או שניים, ואם כשני טוקנים, איך נדע זאת. במקרה הזה, האות הראשונה בשני המילים היא אות גדולה. אולי אפשר להשתמש במידע הזה בכדי להחליט אם ונו רוצים לחלק להתייחס לביטוי כטוקן אחד או שניים (אות גדולה באמצע משפט יכולה לרמז שמדובר בשם).

מה עושים עם תאריכים. תאריכים אפשר לכתוב במספר אופנים, חלקם תלוי במדינה בה כתבו את המסמך. למשל, "3/20/91" או "Mar. 20, 1991" או "20/3/91". כאשר ונו נתקלים במקרים כאלו, ונו צריכים לזהות שמדובר בתאריכים. כמובן שהאפשרות לפורמטים שונים מגדיל את הסיבוכיות במקרה הזה. כמובן, שתאריכים לא מפצלים (למשל בהתאם ל-"/"). כמו כן, ונו רוצים לשמור את כל התארים בפורמט זהה, בכדי שהשאלתא שלנו לא תהיינו מוגבלת לפורמט כלשהו.

במקרים רבים מספרים כוללים בתוכם רווחים, מקפים, ואף תווים אחרים. למשל, My PGP key is B-52,55 B.C., כאשר המספר מייצג מטה-דאטה (למשל תאריך יצירה של הקובץ), ונו נאנדקס אותו באופן נפרד (וואז נוכל לבצע שאילתות חכמות יותר, למשל, הבא לי את כל המסמכים שנוצרו בטווח תאריכים מסויימים).

עד כה דיברנו על בעיות שאנו יכולים להתקל בהם בשפה האנגלית. אבל גם בשפות אחרות ונו יכולים להתקל בבעיות. למשל, בצרפתית, "ה" הידיעה היא המילה Le. כאשר היא מופיעה לפני אות "צליל", נוהגים לקצר ל-"L'". לדוגמא, L'ensemble. מה עושים במקרה זה, האם זה טוקן אחד או שניים? (L' ? L' ? Le ?).

בסינית ויפנית לא קיימים רווחים בין המילים ואף לא בין משפטים. לדוגמא, 莎拉波娃现在居住在美国东南部. 佛罗里达的. החוסר ברווחים מקשה על חלוקה לטוקנים. מעבר לכך, ביפנית קיימים סטים שונים של אותיות, שניתן לשלב יחדיו באותו המשפר. לדוגמא, 500社は情報不足のため時間あた\$500K(約6,000万円). כמו כן, יש ביפנית שימוש בפורמטים שונים, למשל מספרים, תאריכים, יחידות מידה (kg) וכו', שגם הם מקשים על החלוקה לטוקנים.

עברית וערבית נכתבים מימין לשמאל, אולם, הם יכולים להכיל חלקים (למשל מספרים) הנכתבים משמאל לימין. קבצים בפורמטים מסוימים פותר בעיות מסוג זה. למשל, בקובץ המקודד ב-UTF8, כל הטקסט נשמר בכיוון אחד (שמירה פשוטה), ועל התוכנה המציגה את הקובץ מוטלת המשימה של הצגה נכונה של הטקסט.

Terms

ישנן מילים רבות בעלות משמעות סמנטית מועטה (the, a, and, to, be) שמופיעות כמעט בכל מסמך, שניתן להסיר אותן מהמילון. אם ניקח את המילון שלנו, ונסיר ממנו את 30 ה-terms הנפוצים ביותר (לפי ה-collection frequency, שסופר את כל המופעים בכל המסמכים יחדיו, ולא document frequency, שסופר רק את מספר המסמכים שבהם מופיע כל term), נוכל להקטין את המילון בכ-30%. אולם הנטייה כיום היא לא לעשות זאת, וזאת ממס' סיבות:

- קיימים אלגוריתמים לכיוון נתונים מאפשרים לשמור את המילון, הכולל את אותם מילים, בקבצים קטנים יחסית.
- שיטות אופטימיזציה מאפשרות לבצע שאילתות אחזור הכוללות את המילים הנ"ל ללא תוספת משמעותיות בעלויות הביצוע.
- לעיתים, אנו זקוקים למילים הנ"ל, מכוון שהם מהווים חלק ממושגים, "King of Denmark", שמות של שירים, "To be or not to be", "Let it be", וכן הם מציינים קשרים למיניהם, "flights to London".

בשלב הבא אנו צריכים ל"נרמל" את המילים הן בטקסט המאונדקס והן בשאילתות בכדי להביא אותם לאותה הצורה. לדוגמא, USA ו-U.S.A. term הוא צורה "מנורמלת" של טוקן כפי שהיא מופיע במילון של מערכת האחזור שלנו. מילים המופיעות בצורה שונה במסמכים, אך ה-term (המנורמל) שלהם זהה, יהיו באותו posting list. צורה מנורמלת מושגת לרוב בשני אופנים: (1) הורדה של סימני פיסוק, U.S.A ← USA. (2) הורדה של מקפים בין מילים, anti-discriminatory ← antidiscriminatory.

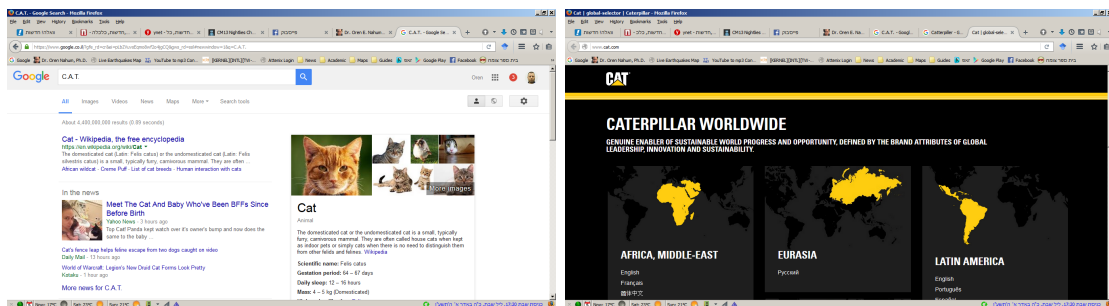
תהליך הנרמול תלוי בשפה שלנו, ולכן בשפות שונות, יתכן ותהליך הנרמול יהיה מעט שונה. בשפות כמו צרפתית אנו נסיר סימנים הקשורים למבטא, résumé ← resume (שימו לב שבאנגלית קיימת המילה resume, שמשמעות שונה, להמשיך, בעוד שבצרפתית משמעות המילה היא קורות חיים). בנרמול מושגים, הכלל הכי חשוב הוא איך המשתמשים נוהגים לכתוב את אותם המושגים, ובהתאם לכך נקבע את האופן שבו הם יופיע במילונים שלנו. לדוגמא, גם בשפות שבהן, הסטנדרט כולל סימנים הקשורים למבטא, המשתמשים נוהגים לכתוב ללא הסימנים הללו.

תהליך הנרמול אינו מוגבל רק למילים. למשל, יש צורך בנרמול של תאריכים, 7/30 vs. 7月30日. במקרה זה אנו רוצים ששני המסמכים יאוחרו, מכוון שמדובר באותו התאריך. הערה: איך אנו יודעים במדובר בתאריך ולא ביטוי מתמטי? ובכן, זה לא פשוט. אנו יכולים להתייחס למידע אחר במסמך בכדי לקבוע במה מדובר. למשל, במסמך שהוא מתמטי באופיו, כנראה שמדובר בביטוי מתמטי.

טוקניזציה ונרמול תלויים בשפה שבה כתוב המסמך, ולכן לעיתים התהליך מקושר לתהליך של זיהוי שפה. יש לשים לב לכך שהן תהליך הטוקניזציה, הנרמול והשאילתות יעשו בהתאם (לפי אותם מאפיינים).

בנוסף, בשפות כמו אנגלית, נהוג להפוך את כל האותיות לאותיות קטנות. במקרים מסוימים לא נעשה זאת, למשל אות גדולה באמצע משפט (שיכולה לציין שם). ברוב המקרים נהפוך את כל האותיות לאותיות קטנות מכוון שהמשתמשים נוהגים לכתוב באותיות קטנות מבלי להתייחס לאופן הנכון שבו יש לרשום את המילים.

תהליך הנורמליזציה יכול לגרום לאובדן מידע. לדוגמא, C.A.T, אשר מתייחס לחברה, הופך ל-cat. במקרה זה, גוגל, לדוגמא, לא יודע למה התכוונו ומחזיר (במקום הראשון) קישור לוויקיפדיה בנושא חתולים.



במקרים מסוימים, במקום לנרמל מושגים נעשה פעולה הפוכה (query expansion). לדוגמא: הקלט: window, החיפוש יתבצע עבור: window, windows. הקלט: windows, החיפוש יתבצע עבור: Windows, windows, window. הקלט: Windows, החיפוש יתבצע עבור: Windows. באופן כללי, אנו מאפשרים חיפוש טוב יותר, עם יותר אפשרויות. מצד שני, אנו מבצעים יותר חיפושים (זמן חיפוש) ומחזירים יותר תוצאות.

כיצד אנו יכולים לטפל במילים נרדפות? נניח שהשאלתא היא car, יתכן שאנו רוצים לקבל מסמכים המכילים גם automobile. באופן דומה, נניח שהשאלתא היא color, אנו רוצים לקבל גם מסמכים בהן כתוב colour.

באופן ידני אנו יכולים ליצור רשימה של מילים נרדפות. במקרים כאלו יש לנו שתי אפשרויות. הראשונה, אנו מבנה שאלתא מקבילה, שבה נחפש עבור כל מילה את כל המילים הנרדפות שלה, למשל השאלתא color תהפוך ל-color or colour, ועבור השאלתא car נקבל את השאלתא car or automobile. אפשרות שניה, לאנדקס מילים כאלו מספר פעמים, בהתאם למילים הנרדפות שלהם.

שגיאות כתיב - אחת הדרכים להתמודד עם שגיאות כתיב היא ע"י שימוש בסאונדקס. סאונדקס הוא תחליף פונטי למילה, כך שלשתי מילים הנשמעות אותו הדבר יש את אותו הסאונדקס. אלגוריתם סאונדקס המופעל על string מסוים, בודק אותו, מזהה את העיצורים השונים שיש בו, ובונה string חדש שבו יש ייצוג פונטי של ה-string המקורי. עבור שני string-ים שנשמעים אותו הדבר, אלגוריתם הסאונדקס יצור string פונטי זהה. השימוש בסאונדקס רלוונטי בעיקר עבור שמות.

Lemmatization & Stemming

Lemmatization הינו התהליך של העברת מילים לצורת הבסיס שלהם. לדוגמא:

- .am, are, is → be
- .car, cars, car's, cars' → car
- .the boy's cars are different colors → the boy car be different color

בכדי לבצע lemmatization אנו צריכים מנגנון שמכיר את השפה שלנו.

Stemming גם הוא התהליך של העברת terms (מושגים/תארים) לצורת הבסיס. הפעולה נעשית ע"י הסרה של תחליות או סיומות מה-terms שלנו. לדוגמא:

- automate(s), automatic, automation → automat

פעולת ה-stemming אינה, בהכרח, מייצרת לנו מילים "חוקיות".

האלגוריתם הנפוץ ביותר ל-Stemming באנגלית הוא האלגוריתם של פורטר. תוצאות האלגוריתם לא נופלות באיכותן מהתוצאות המתקבלות מהאלגוריתמים האחרים (נבדק ע"י השוואת תוצאות). האלגוריתם מורכב מחמישה שלבים לקיצור המילים והעברתם לצורת הבסיס, המבוצעים אחד אחר השני. כל שלב כולל מספר כללים לבחירת הפעולה של אותו שלב, לדוגמא, בחירת הפעולה שפועלת על הסיומת הארוכה ביותר.

- sses → ss
- ies → I
- ational → ate
- tional → tion

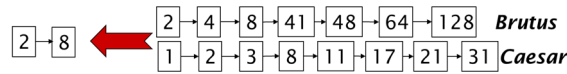
כללים אחרים משתמשים במושג של "משקל" או "אורך" (ביחס למס' הברות), בכדי לקבוע אם ניתן להפעיל את הכלל על המילה או לא.

- (m>1) EMENT → _ (מילה שמסתיימת ב-ement ומה שמופיע לפנייה הוא יותר מתו אחד)
 - replacement → replac
 - cement → cement

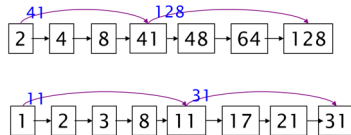
כל הכללים שדיברנו עליהם תלויים בשפה שאייתה אנו עובדים. Stemming לא תמיד עוזר לנו, לרוב זה תלוי גם בשפה שלנו.

Inverted Index & Skip pointers

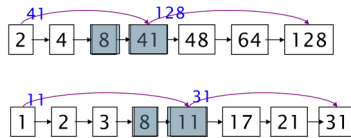
נניח שנתונה השאילתא Brutus and Caesar. בכדי לענות עליה, אנו עוברים על שני posting lists בו זמנית. זמן הריצה הוא לינארי, ותלוי במספר המסמכים בכל רשימה. נניח שמספר המסמכים ברשימה הראשונה הוא n , וברשימה השנייה m , אז זמן הריצה הוא $O(n+m)$.



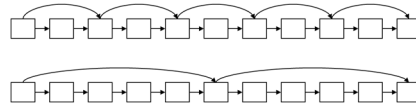
האם ניתן לעשות זאת מהר יותר? כן. ניתן לשפר את זמן הריצה ע"י הוספת מצביעים ה"קופצים" (skip pointers) לנקודות שונות ברשימה שלנו. היכן עלינו למקם את המצביעים הללו? לאן הם צריכים להצביע? ואיך אנו משתמשים בהם?



נניח ועברנו על הרשימות שלנו והגענו ל-8 בשתייהן. אם נמשיך לעבור על שתי הרשימות הרי שנקבל 41 עבור הרשימה הראשונה ו-11 עבור הרשימה השנייה. ל-11, הערך הנמוך, קיים מצביע "קופץ". אם נשתמש בו, נגיע ל-31, שעדיין נמוך מ-41. כלומר אנו יכולים לדלג ישירות ל-31, מבלי לפגוע בתהליך האיחוד.



ככל שיש יותר מצביעים, המצביעים "קופצים" למרחקים קצרים יותר, והסיכוי ל"קפיצות" גדל. פחות מצביעים, המצביעים "קופצים" למרחקים ארוכים יותר, והסיכוי ל"קפיצות" קטן.



בכדי לקבוע כמה והיכן מצביעים "קופצים" לשים, נשתמש בהיוריסטיקה פשוטה: עבור רשימה בגודל L , נשתמש ב- \sqrt{L} מצביעים "קופצים", הממוקמים במרחקים שווים. היוריסטיקה זו מתעלמת מהתפלגות המושגים בשאילתא. היוריסטיקה זו קלה למימוש אם אנו לא משנים את הרשימות שלנו. אם הרשימות משתנות בתדירות גבוהה (עדכונים), המימוש הוא בעייתי.